

REMARKS

Applicant would first like to thank Examiner Daniel L. Hoang for this examination.

Claims 1-14 are currently pending in the application, and claims 1-14 stand rejected under 35 USC 102 as being anticipated by US Pat. No. 6,823,460 to Hollander et al.

CLAIM REJECTIONS - 35 USC 102

Claims 1-14 stand rejected under 35 USC 102 as being anticipated by 6,823,460 to Hollander et al. (hereafter, Hollander).

CLAIMS 1, 14

Claims 1 and 14, as originally filed are allowable because they include features that are neither disclosed nor suggested by Hollander or any other references, either individually or in combination.

“in response to a system call, executing a hook routine at a location of said system call,”

Claim 1 includes that feature “in response to a system call, executing a hook routine at a location of said system call,” and claim 14 includes the similar element “program instructions responsive to a system call for executing a hook routine at a location of said system call,” which are neither disclosed nor suggested by Hollander or any other reference.

As described in the present application, in an exemplary embodiment a system call is an operation which transfers control of a processor, such as by stopping the current processing in order to request a service provided by an interrupt handler. The interrupt handler in turn is a program code at a memory location identified by an interrupt vector table. In this example, the system call comprises an interrupt. By use of the interrupt vector table, the system executes the software routine located at the memory location designated (or pointed to) for the particular

interrupt in the interrupt vector table. In claim 1, in response to a program making a system call during execution of the program, the processor is pointed to a memory location for the system call. The hook routine, as provided in claim 1, is located at the location pointed to by the system call, and is executed in response to the system call. This is significant because the programming level at which software interrupts work is the level at which many computer viruses work (see specification page 8, line 30 to page 9, line 1). Moreover, the present application clearly indicates that it is desirable to hook the lowest level calls where possible (see specification page 9, lines 4-5).

Hollander does not disclose or suggest executing a hook routine in response to a system call. In fact, Hollander defines hooking for the purpose of interpreting their specification as obtaining control of a desired API (col. 8, lines 37-39). Moreover Hollander does not execute a hooking routine in response to a system call, but rather overwrites a section or sections of the API to control the code behavior in response to the API being called (col. 8, lines 40-44). Applicants respectfully contend that overwriting the API does not disclose or suggest executing a hook routine at a location of the system call as a patch is not a hook routine.

The Examiner appears to argue that this feature is disclosed by Hollander at col. 6, lines 7-11. Applicants respectfully disagree. The cited text provides that an API Interception Control Server monitors the host operating system for system calls through a system call interception component. Hollander does not suggest a hook routine at the location of the system call or any hook routine. Detection of a system call is not the same as executing a hook routine in response to a system call. Moreover, Hollander is silent as to how the interception component operates and the Examiner may not assume a method to establish anticipation. It is not sufficient that a certain thing may result from a given set of circumstances (see *In Re Robertson*, 169 F.3d 743, 745, 49 USPQ2d 1949, 1951 (Fed. Cir. 1999)(citations and internal quotes omitted).

The Examiner has failed to make a prima facie case that the element "in response to a system call, executing a hook routine at a location of said system call," is anticipated by

Hollander.

“determine a dataflow or process requested by said call”

Claims 1 and 14 include a second element “determine a dataflow or process requested by said call,” which Applicant contend are neither disclosed nor suggested by Hollander or any other reference.

In the present application, in response to a system call being made, the hooking routine executes to monitor and display the operation of the computer system (see specification page 10, lines 1-8). The hooking routines generate an icon or other graphical representation of the current operation to be performed by the original routine (i.e., the routine called by the system call). Thus, in claims 1 and 14, the hooking routine determines the actual function to be performed by the system call (e.g., transferring data, mathematical function, deleting data, copying data, etc.). As pointed out previously, it is important to monitor the system call functions, as this is where many malicious software programs operate.

The Examiner appears to argue that Hollander discloses this feature at col. 6, lines 7-11, discussed above. Hollander does not disclose or suggest any hook routine or determining a data flow or process requested by a system call, much less a hook routine that determines a data flow or process requested by the call.

The Examiner has failed to make a prima facie case of anticipation of the second element, “determine a data flow or process requested by said call.”

“determine another data flow or process for data related to that of said call”

Claims 1 and 14 include a third element, “determine another data flow or process for data related to that of said call,” which Applicant contend are neither disclosed nor suggested by

Hollander or any other reference.

The present invention provides for associating a current system call function with another system call function by matching file names or memory locations, for example. This step is important to creating a meaningful process flow to track a data flow or process, as operations that in isolation may not appear malicious, but viewed together, show a malicious pattern. Malicious software may manipulate data, for example, using a series of steps which individually do not appear malicious. By following the data or process flow the danger, however, becomes apparent.

Hollander does not disclose or suggest stringing together related system call operations to track data flow or process flows. The Examiner apparently argues that this feature is provided by Hollander at col. 6, lines 12-20. Applicants respectfully disagree. The cited text merely recites that an API interception control server monitors a host operating system for system calls and takes an action according to the type of system call. Hollander is silent as to how the monitoring is performed. Moreover, Hollander does not disclose or suggest any determination of a data or process flow or determination of related data or process flow, much less stringing together related data or process flows.

The Examiner has failed to make a prima facie case of anticipation of the third element, "determine another data flow or process for data related to that of said call."

"automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process"

Claims 1 and 14 include a fourth element, "automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process," that is neither disclosed nor suggested by Hollander or any other reference.

This element is directed to the hooking program combining the data flow of a current system call with the data flow of related system calls and presenting this data flow information in

a specific useful form, namely a consolidate information flow diagram. A diagram is a graphical representation, as shown in Figs. 3, 4, and 5 and described in the specification at pages 11-15.

Hollander does not disclose or suggest generating a consolidated information flow diagram. The Examiner argues that Hollander discloses this feature in Fig. 7, step 156 contending that an API flow table is analogous to a consolidated information flow diagram. Applicants respectfully disagree. Hollander does not disclose or suggest a data or process flow diagram that consolidates data or process flow over multiple system calls. Moreover, Applicants respectfully content that the API Flow Structure Table is not defined in Hollander, and that Hollander is unclear regarding what the API Flow Structure Table is and how it functions.

The Examiner has failed to make a prima facie case of anticipation with respect to the fourth element, "automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process."

CLAIM 8

Claim 8, as originally filed is allowable because it includes features that are neither disclosed nor suggested by Hollander or any other references cited in the First Office Action, either individually or in combination.

elements previously discussed under claim 1

"means responsive to a system call, for executing a hook routine at a location of said system call"

This element is directed to a computer operating a program of instruction substantially similar to the first element argued under claims 1 and 14. These arguments will not be repeated here.

“determine a dataflow or process requested by said call”

This element is directed to a computer operating a program of instruction substantially similar to the second element argued under claims 1 and 14. These arguments will not be repeated here.

“determine another data flow or process for data related to that of said call”

This element is directed to a computer operating a program of instruction substantially similar to the third element argued under claims 1 and 14. These arguments will not be repeated here.

“automatically generate a consolidated information flow diagram showing said data flow or process of said call and said other data flow or process”

This element is directed to a computer operating a program of instruction substantially similar to the fourth element argued under claims 1 and 14. These arguments will not be repeated here.

“means for displaying said information flow diagram”

Claim 8 includes the element “means for displaying said information flow diagram.” The structure associated with this element is the display screen 120. Thus, the information flow diagram representing the information flow determined by computer 110 is displayed on display screen 120. Hollander does not describe either an information flow diagram or a display screen. Accordingly, the Examiner has failed to make a prima facie case of anticipation.

CLAIM 2

Claim 2, as originally filed is allowable independently of its parent claim 1, because it includes features that are neither disclosed nor suggested by Hollander or any other references cited in the First Office Action, either individually or in combination.

“a user monitors said information flow diagram”

Claim 2 includes the element “a user monitors said information flow diagram.” Hollander does not disclose or suggest a user monitoring an information flow diagram. A user monitoring the information flow diagram is significant because it allows the user to detect suspicious activity in real time and take remedial measures (page 15 lines 13-25).

The Examiner appears to argue that this feature is provided by Hollander at col. 2, lines 45-52. Applicants respectfully disagree. This text does not provide a user or a flow diagram, much less a user monitoring an information flow diagram.

The Examiner has failed to make a prima facie case of anticipation with respect to the fourth element, “a user monitors said information flow diagram.”

CLAIMS 3 and 9

Applicants have contended that claims 3 and 9, as originally filed are allowable independently of their parent claims 1 and 8, respectively, because they include a feature that is neither disclosed nor suggested by Hollander or any other references cited in by the Examiner, either individually or in combination.

“said information flow diagram illustrates locations of said data at stages of a processing activity”

Claims 3 and 9 include the element "said information flow diagram illustrates locations of said data at stages of a processing activity." The present invention provides an embodiment in which the flow of data is captured and the location of the data at stages of processing is illustrated. This allows a user to quickly see where data is being transferred to, and therefore, whether data is being manipulated in an undesirable way. Hollander does not disclose or suggest illustrating locations of data during processing.

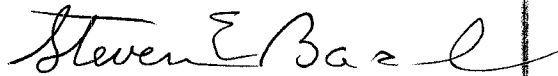
The Examiner appears to argue that this feature is disclosed by Hollander in Fig. 3, elements 154-165. Applicants respectfully disagree. Fig. 3 does not include elements 154-165. Moreover, Fig. 3 is directed to injecting an API Interception Module into address space of an active process (see col. 6, lines 40 et. Seq.) and not to a flow diagram that illustrates the location of specific data during different phases of processing.

The Examiner has failed to make a prima facie case of anticipation with respect to the fourth element, "said information flow diagram illustrates locations of said data at stages of a processing activity."

CONCLUSION

In view of the amendments and arguments presented herein, Applicant respectfully contends that claims 1-15 are in condition for allowance. Accordingly, Applicant respectfully requests entry of the amendments, reconsideration and allowance of claims 1-15 and issuance of letters patent.

Sincerely,

A handwritten signature in cursive script that reads "Steven E. Bach". The signature is written in dark ink and is positioned above the printed name and title.

Steven E. Bach
Attorney for the Applicant
Reg. No. 46,530